Handout 17

Functions

17.1 A first example

```
/*
first program with user-defined function
*/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

// function that prints "hello?"
void hello()
{
  printf("hello?\n");
}

// the main function
int main()
{
  hello();
  hello();
  hello();
}
```

Use this to make a loop that prints `hello?` 10 times.

17.2 Functions can take inputs

```
/*
first program with user-defined function
that takes inputs
*/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

// print out number
void youEntered(int number)
{
  printf("You entered %d\n", number);
}

// main function
int main()
{
  int input;
  printf("Give me an integer:\n");
  scanf("%d", &input);
  youEntered(input);
}
```

Modify the code above so that the user gives two inputs and the function reports the sum of the inputs. You will want to use the following format for the user-defined function

```
void reportSum(int number1, int number2)
{
  ?????
}
```

17.3 Functions can give outputs

```
/*
first program with user-defined function
that takes inputs
*/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

// get an integer
int getInput()
{
  int input;
  printf("Give me an integer:\n");
  scanf("%d", &input);
  return input;
}

// print out number
void report(int number)
{
  printf("You entered %d\n", number);
}

// main function
int main()
{
  int num;
  num = getInput();
  report(num);
}
```

Modify the function `getInput()` so that it requires the user to enter a positive integer. If the user enters a negative integer, then the function should ask the user for a different input.

Note: There is an important programming lesson here. By putting the "get input" process in to a separate function, we can deal with all of the various special cases, such as making sure that the input is valid, in one spot that is "isolated" from the rest of the program.

17.4 Homework: `fibonacci.c`

Complete the following code to obtain a program that computes Fibonacci numbers.

```
// computes fibonacci numbers
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int fib(int n) // compute nth fibonacci number for n>2
{
  ?????
}

int getInput() // get integer from user
{
  int input=0;
  while(input <=0)
  {
    printf("Give me a positive integer:\n");
    scanf("%d", &input);
    if (input <=0)
    {
      printf("The integer needs to be positive\n");
    }
  }
  return input;
}

void report(int output) // print results
{
  printf("The corresponding Fibonacci number is %d\n",output);
}

int main()
{
  int n;
  n = getInput();

  if (n==1 || n==2) // special cases
  {
    report(1);
  }
  else // general case
  {
    report(fib(n));
  }
}
```

17.5 How to put the main function first

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
  // declare the other functions
  int getInput(void);
  void report(int); // or: void report(int temp);

  // the actual main function
  int num;
  num = getInput();
  report(num);
}

int getInput(void) // get an integer
{
  int input;
  printf("Give me an integer:\n");
  scanf("%d", &input);
  return input;
}

void report(int number) // print out number
{
  printf("You entered %d\n", number);
}
```

17.6 Challenge: `recursive-fib.c`

It is possible for a function to refer to itself; a function that uses this feature is called a *recursive function*. For example, one can compute factorial recursively using the following function.

```
int fact(int n)
{
  if (n==1)
    return 1;
  else
    return n*fact(n-1);
}
```

Can you define a recursive function to compute Fibonacci numbers? What other programs have we written in this class that can be framed using recursive function?